(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 014 627 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.06.2000 Bulletin 2000/26

(51) Int Cl.7: **H04L 12/56**

(21) Application number: **99309837.5**

(22) Date of filing: **07.12.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **22.12.1998 US 218576**

(71) Applicant: **LUCENT TECHNOLOGIES INC.**
Murray Hill, New Jersey 07974-0636 (US)

(72) Inventors:
- **Kodialam, Muralidharan Sampath**
  Malboro, New Jersey 07746 (US)
- **Lau, Wing Cheong**
  Eatontown, New Jersey 07724 (US)
- **Yan, Anlu**
  Eatontown, New Jersey 07724 (US)

(74) Representative:
Johnston, Kenneth Graham et al
Lucent Technologies (UK) Ltd,
5 Mornington Road
Woodford Green Essex, IG8 OTU (GB)

(54) Constrained shortest path routing method

(57) A real-time method for routing subject to an acceptable delay constraint between nodes in high-speed data networks, such as PNNI protocol networks, uses an illustrative two-phase algorithm employing Dijkstra's algorithm at each phase. In an illustrative first phase, the Dijkstra SPF algorithm is used in seeking the shortest cumulative delay from the destination to the source, thereby generating cumulative delay labels from a node *j* to the destination node k. The delay results are then employed in the second phase, where the Dijkstra SPF algorithm is illustratively employed for determining administrative weight (AW) as the link metric subject to modification in accordance with results obtained in the first phase.

EP 1 014 627 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Field of the Invention

[0001] The present invention relates to methods and systems for routing data paths in high-speed data networks. More particularly, the present invention relates to methods and systems for selecting preferred routing paths in such networks employing private network node interface (PNNI) protocols.

Background of the Invention

[0002] High capacity data networks, such as those using high-speed asynchronous transfer mode (ATM) switches offer promise not only of high-speed data delivery, but also a variety of quality of service (QoS) guarantees. An important element of establishing paths interconnecting nodes in a such high-speed data networks is an efficient, reliable routing mechanism.

[0003] Recently, the ATM Forum has standardized on certain PNNI protocols. As defined, these protocols provide for signaling and routing protocols that permit connection setup and release with multiple QoS parameters between nodes. These protocols also provide for exchange of topology state information between nodes. See further, "Traffic Management Specification," *ATM Forum*, 95-0013, R10, Feb. 1996; and Dykeman, D., and M. Goguen, "PNNI Specification," Ver. 1.0, *ATM Forum*, AF-PNNI-0055.000, May, 1996.

[0004] Many implementations of the PNNI protocols have been proposed, with a variety of performance and QoS features. An important consideration in many networks, such as those using the PNNI protocols, is interconnecting desired nodes while employing a minimum of network resources. In particular, it is desired to interconnect nodes using the shortest interconnecting paths. Additionally, it has become ever more important - especially for many time-dependent applications (such as multimedia applications) - to seek to achieve low or minimum delay time for transmission between selected nodes. QoS criteria based on such reduced or minimum delay are therefore of great importance for critical applications, and provide important competitive differentiation for service providers.

[0005] There are many solutions for selecting the shortest path (or minimum defined cost) between selected nodes, even in a network having a large number of interconnected nodes. Prominent among these shortest path algorithms is the well-known Dijkstra algorithm. Solutions based on the Dijkstra and other algorithms may often be modified to permit the inclusion of certain conditions relating to various QoS criteria. Experience with Dijkstra algorithm solutions and corresponding coding implementations have proven reliable in many cases and have been adopted for real-time network implementations. It is desirable, therefore, to reuse existing code and maintain functional compatibility between delay-constrained solutions and existing shortest path implementations. As is well known in the art, however, network solutions of the shortest path problem subject to an additive delay condition is a so-called "NP hard" problem that proves intractable in real-time network contexts using current implementation technologies.

Summary of the Invention

[0006] The limitations of the prior art are overcome and a technical advance is made in the network routing arts in accordance with illustrative embodiments of the present invention, as described in the following detailed description.

[0007] In accordance with an illustrative embodiment of the present invention, determinations are made at a source node in a network of the short weight paths to each other node in the network, subject to satisfying an acceptable delay constraint. Advantageously, these determinations are performed in a two-phase method employing a modified Dijkstra's algorithm at each phase.

[0008] In an illustrative first phase, the Dijkstra SPF algorithm is used in seeking the shortest cumulative delay from the destination to the source, thereby generating cumulative delay labels from a node *j* to the destination node *k*. The delay results are then employed in the second phase, where the Dijkstra SPF algorithm is illustratively employed for determining administrative weight (AW) as the link metric subject to modification in accordance with another aspect of the present invention. More specifically, in an illustrative embodiment it proves advantageous to (i) label a node *i* with cumulative AW from the source node, (ii) track cumulative delay ($D_{s,i}$) from the source during forward AW labeling, and (iii) labeling neighboring nodes *j* of a permanently labeled node *i* only if ($D_{s,i} + d_{j,k} + \delta_{i,j}$) is less than the specified end-to-end delay constraint (where $\delta_{i,j}$ is the delay for the link between nodes *i* and *j*).

Brief Description of the Drawing

[0009] The present invention as described in illustrative embodiments in the following detailed description and in the attached drawing, wherein:

[0010] FIG. 1 shows an example network for illustrating an illustrative embodiment of the present invention.

[0011] FIG. 2 is useful in understanding Phase 1 of an illustrative embodiment of the present invention as applied to the example network of FIG. 1.

[0012] FIG. 3 is useful in understanding Phase 2 of an illustrative embodiment of the present invention as applied to the example network of FIG. 1.

Detailed Description

[0013] FIG. 1 shows an illustrative network of nodes corresponding, for example, to the nodes in an ATM network. Each source (originating) node includes a control processor for receiving network node state information (as is well-known) and determining a low-cost or near-optimum shortest path analysis, subject to a maximum cumulative delay constraint. Costs are illustratively measured, in part, in terms of administrative weight (AW). Additionally, in an accordance with another aspect of the present invention, costs are also measured, in part, in terms of link delay. Initially, the present discussion proceeds in terms of a single source seeking a low-cost routing to a single destination (*unicast* operation). Single-source, multi-destination (*multicast*) operation will be discussed below.

[0014] Routing is performed, for a given call, at the source or on behalf of the source by an associated separate or shared processor. In accordance with PNNI protocols, link state parameters communicated between the nodes are of two types: *link metrics* and *link attributes*. A link metric is a state parameter for a link, which parameter is a combination of all link parameters along a given path. Thus, such a link metric can indicate whether a particular path is acceptable for a given connection. A link attribute, on the other hand, is a link state parameter that reflects the state of a given link in a path. Path delay is a link metric suitable for use in path weight determinations in accordance with one aspect of the present invention (rather than AW). Path delay is taken as the sum of the delay of all of the links in a path. Received path delay information from other network nodes is advantageously stored in table form at a node processor.

Model

[0015] It proves useful to more precisely characterize network routing solutions in terms of a network model. For this purpose, let $G = (V, E)$ represent a directed graph representation of a network. We assume that there are n nodes and m arcs in the tree. Let $s \in V$ represent the source and t represent the destination. Let $(i, j) \in E$ denote a directed edge (arc) in the graph. Let $w_{ij} \geq 0$ represent the weight of arc (i, j) and $d_{ij} > 0$ represent the delay incurred along arc (i, j). Let $P(i, k)$ represent the set of simple paths from node i to node k. The weight of a path $P \in P(i,k)$, denoted by $w(P)$, is defined as the sum of the weight of the arcs in the path. The delay of a path $P \in P(i, k)$, denoted by $d(P)$, is the sum of the delay of the arcs in the path. The objective of the routing problem is to find the path of minimum weight from source s to destination t such that the delay of that path is less than some pre-specified value D, *i.e.*,

$$\min_{P \in P(s,t)} w(P)$$

subject to

$$d(P) \leq D.$$

As noted above, this problem is NP-hard. Next, we develop a heuristic algorithm that seeks to minimize the weight of the path while avoiding violating the delay constraints. We show that the algorithm always terminates with a feasible path (with respect to the delay constraints) if one exists. We will also show in the sequel that implementations of the present inventive algorithm obtain paths of low weight.

Algorithm Description

[0016] In this section, we describe the algorithm that is run at each source. We first describe the algorithm for a particular destination. This algorithm is then used as a subroutine to compute short weight path to all the destinations from a given source.

Single Sink

[0017] Let s represent the source node. In this section, we determine a short weight path from s to a given destination

node *t*. The processing of the algorithm at a node processor advantageously proceeds in two phases. Phase 1 of the algorithm involves running the well-known Dijkstra algorithm for paths from the destination to the source using the delay labels as the length of the arc. The distance labels (in delay) that result from the execution of Phase 1' are used in Phase 2 of the algorithm. We now give the algorithm for Phase 1.

PHASE 1 (s, t, d)

- INITIALIZATION
  - $U = V$, $W = \phi$, $\delta_t = 0$, $\delta_j = \infty \; \forall_j \neq t$
- ITERATIVE STEP
  - $k = \text{Arg min}_{j \in U} \delta_j$
  - $U = U \backslash \{k\}$ and $W = W \cup \{k\}$.
  - For each $j \in U$, $(j, k) \in E$, if $\delta_k + d_{jk} < \delta_j$ then $\delta_j = \delta_k + d_{jk}$
- TERMINATION TEST
  - Stop if $s \in W$.

[0018] Note that δ_j represents the shortest path length from node j to the destination t. If $\delta_s > D$ then there is no feasible solution to the problem and the algorithm terminates. If $\delta_s \leq D$, then Phase 2 of the algorithm, which is a modified version of Dijkstra's algorithm, is executed from the source s.

[0019] For the ease of the algorithm description, we define the generalized weight of an edge (i,j) as

$$v_{ij} \triangleq (w_{ij}, d_{ij}) \qquad (1)$$

The generalized weight of a simple path $P \in P(i, k)$ is defined as $v(P) \bullet (w(P), d(P))$.

[0020] In Phase 2 the predecessor of each node is maintained in $\pi[i]$, $\forall i \in V$, and a two-tuple label (ω, ψ) is maintained for each node, where ω is the accumulated weight from the source s along a feasible path and ψ is the delay from the source s along the same path. A lexicographic order is defined for these two-tuples, namely,

Definition 1 Define

$$(\omega_i, \psi_i) < (\omega_j, \psi_j)$$

if

- $\omega_i < \omega_j$, *or*
- $\omega_i = \omega_j$ *and* $\psi_i < \psi_j$

[0021] Phase 2 of the algorithm is:

PHASE 2 (s, t, w, d)

- INITIALIZATION
  1: $U = V$, $W = \emptyset$, $\omega_s = \psi_s = 0$, $\omega_j = \psi_j = \infty \; \forall_j \neq s$,
  $\pi[j] = NULL \; \forall j \in V$
- ITERATIVE STEP
  2: $k = \text{Arg min}_{j \in U} (\omega_j, \psi_j)$
  3: $U = U \backslash \{k\}$ and $W = W \cup \{k\}$.
  4: 4.1: For each $j \in U$, $(k, j) \in E$

4.2: if $(\omega_k + w_{kj}, \psi_k + d_{kj}) < (\omega_j, \psi_j)$
4.3: if $(\psi_k + d_{kl} + \delta_j \leq D)$
4.4: $\omega_j = \omega_k + w_{kj}$
4.5: $\psi_j = \psi_k + d_{kj}$
4.6: $\pi[j] = k$

- TERMINATION TEST
  5: Stop *if* $t \in W$.

[0022] Note that if line 4.3 is taken out, the above algorithm is just Dijkstra's algorithm with respect to the generalized weight v. In other words, it is Dijkstra's algorithm with respect to the weight with the delay label ψ used as the tie-breaker. Since all the arguments used in proving Dijkstra's algorithm still apply when we use the generalized weight v defined in 1, we have the following lemma.

Lemma 1 If we run Dijkstra's algorithm on *G* with respect to the generalized weight v, then at termination,

$$(\omega_i, \psi_i) = \min_{P \in P(s,i)} v(P)$$

[0023] The following theorem summarizes the properties of our algorithm.

Theorem 2 Phase 2 terminates with a path that satisfies the delay constraint. In addition, if there are shortest paths with respect to weight that satisfy the delay constraint, the algorithm will find one.

Proof.

[0024] First we show that when $\delta_s \leq D$, Phase 2 will find a path that satisfies the delay constraint.

[0025] From line 4.3 and 4.4, one can see that the weight label of a node *i*, ω_i, assumes a finite value only when node *i* is on a feasible path (*i.e.*, there is at least one path from *i* to *t* that satisfies the delay constraint). In addition, line 3 removes one element from set *U* in each iteration. Therefore, if the algorithm fails to find a feasible path, we must have $\omega_i = \infty$, $\forall i \in U$ at some point during the execution of the algorithm. This is not possible due to the following reason: Whenever a node is included in the set *W* there exists a path from the source to the destination through that node that satisfies the delay constraint. Since $s \in W$ and $t \in U$ (otherwise the algorithm has succeeded), $\exists k \in W, j \in U$, s.t. edge (k, j) is on a feasible path. When *k* was included in *W* the value of w_j would have been updated to a finite value and hence a contradiction.

[0026] To prove the second statement of the theorem, first consider running Dijkstra's algorithm with respect to the generalized weight v. When it terminates with a simple path from *s* to *t*, denoted by $P^*(s,t) = (s, i_1, \ldots, i_{k_0} = t)$, each node on this path has a label $(\omega_{ik}, \psi_{ikk})$, $k = 1, \ldots, k_0$. We know from Lemma 1 that $P^*(s, t)$ is the shortest path with respect to weight among all the paths from *s* to *t*, and if there are more than one such shortest paths, $P^*(s,t)$ has the smallest delay among them. Therefore, if there are shortest paths with respect to weight that satisfy the delay constraint, $P^*(s, t)$ is one among them.

[0027] Let's assume for now that no other shortest path with respect to weight has the same delay as $P^*(s, t)$. We then use mathematical induction to prove that upon termination of Phase 2, a) the label for node i_k *on* $P^*(s, t)$ is still (ω_{ik}, ψ_{ik}); and b) the path from *s* to *t* will be found to be $P^*(s, t)$.

[0028] Since $P^*(s, t)$ is a feasible path, the condition on line 4.3 will be satisfied when are (s, i_1) is explored. By Lemma 1, arc are (s, i_1) has the smallest generalized weight v_{si_1}, among all the paths from *s* to i_1, so node i_1 will be permanently labeled as $(\omega_{i_1}, \psi_{i_1}) = (w_{si_1}, d_{si_1})$ and its predecessor will be *s*.

[0029] Note that it is possible that there are other paths from *s* to i_1 whose generalized weight is the same as (w_{si_1}, d_{si_1}), in which case node i_1 may be permanently labeled not from s but from the last node on one of these paths, say the last node *of* $P'(s, i_1)$. But since the value of the label for i_1 will be the same, and *s* will still be on the predecessor tree of i_1, possibly with some intermediate predecessor nodes, finding path $P'(s, i_1)$ is no different from finding (s, i_1) as far as weight and delay are concerned. Therefore, we will ignore such cases without loss of generality in the rest of the proof.

[0030] Since node i_1 is directly connected to s, it acquires its permanent label when s is taken out of set U, which happens during the first iteration of the algorithm. Because the label values do not increase and $(\omega_{ik}, \psi_{ik}) \leq (\omega_{ik+1}, \psi_{ik+1})$ for $k = 1, \ldots, k_0 - 1$, we know that if $(\omega_{i_1}, \psi_{i_1}) < (\omega_{i_2}, \psi_{i_2})$ (or equivalently $(w_{i_1 i_2}, d_{i_1 i_2}) > (0,0)$) then node i_1 will be removed from set U before $i_2, \ldots, i_{k_0}$. If $(\omega_{i_1}, \psi_{i_1}) = (\omega_{i_2}, \psi_{i_2})$ (*i.e.* $(w_{i_1 i_2}, d_{i_1 i_2}) = (0,0)$), then it is possible that i_2 is removed before i_1 from set *U*. But this simply means that we have found another path from *s* to i_2 with the same weight and delay as $P^*(S, i_2) = (s, i_1, i_2)$. Again, we don't distinguish such cases from that of i_1 being removed from set *U* before i_2, for these

paths are equivalent as far as weight and delay are concerned. For the rest of the proof, we assume $(\omega_{ik}, \psi_{ik}) < (\omega_{ik+1}, \psi_{ik+1})$ without loss of generality.

[0031] Assume for the purpose of induction that the nodes $i_1,..., i_k$ have been moved to set W and permanently labeled as $(\omega_{i_1}, \psi_{i_1}),..., (\omega_{ik}, \psi_{ik})$. Further assume that the path from s to i_k is found to be $(s, i_1,..., i_k)$ and i_k is removed from set U before nodes $i_{k+1},...,i_{k0}$. *We* now show that node i_{k+1} *will* be the next on the path and will be labeled $(\omega_{ik+1}, \psi_{ik+1})$.

[0032] When node i_k is removed from set U, arc (i_k, i_{k+1}) is explored. Since the path from s to i_k is part of $P^*(s, t)$ and i_{k+1} is on $P^*(s, t)$, condition on line 4.3 is satisfied. Hence i_{k+1} will be labeled as $(\omega_{ik}, \psi_{ik}) + (w_{ikik+1}d_{ikik+1}) = (\omega_{ik+1}, (\psi_{ik+1})$, and its predecessor will be i_k. Because $(\omega_{ik+1}, \psi_{ik+1})$ is the smallest label by Lemma 1, node i_{k+1} will not be relabeled. In addition, i_{k+1} will be removed from set U before nodes $i_{k+2},..., i_{k0}$ because it acquires its permanent label when node i_k is removed from set U and this label is smaller than those of $i_{k+2},...,i_{k0}$. This concludes our induction.

[0033] Finally, if there are other shortest paths with respect to weight that has the same delay as P* (s, t), then it is possible that such an alternative path is found instead *of* $P^*(s, t)$. But these paths are equivalent as far as weight and delay are concerned.

Multiple Sink Case

[0034] The objective in PNNI is to determine the shortest path from source s to all other nodes. This is done by executing Phase 1 and Phase 2 from the source to each sink. However, when the algorithm is executed to determine a short weight path from the source to the destination t_1, it might give information about short paths from the source to other destinations. In particular, for all destinations in the set F, the optimal solution to the constrained shortest path problem is known. This is exploited in the multiple sink case. The algorithm multiple sink is given below:

MULTIPLE DESTINATIONS(s)

[0035]

- INITIALIZATION
  - $G=V$ $F=\emptyset$ $M=\infty$.
- ITERATIVE STEP
  - Pick $t \in G$
  - PHASE 1 (s, t, d)
  - PHASE 2 (s, t, w)
- TERMINATION TEST
  - Stop if $G \in \emptyset$ else go to the ITERATIVE STEP.

[0036] Returning now to the example network of FIG. 1, we consider the application of phase 1 processing described above. As illustrated in FIG. 2, phase 1 comprises determining the smallest delay labels $(d_{j,k})$ for the delay from a node j to the destination k using the reverse Dijkstra SPF algorithm. It is assumed that the Source-to-Destination delay is constrained to be less than or equal to 10 units. The link parameters are presented as (link AW, link delay). So, for example, for the topmost path in FIG. 2, the link (AW, delay) tuple is (1,4).

[0037] In FIG. 3, the results of phase 1 processing are applied in phase 2 processing. In particular, we again run the Dijkstra SPF, this time from the source. AW is used as the link metric for the "shortest" path determination, but (i) each node is labeled with cumulative AW sum from the source, (ii) cumulative delay from the source (D_{si}) labels of a neighboring node j of a permanently labeled node i are updated only if $(D_{s,i} + d_{j,k} + \delta_{i,j})$ meets the end-to-end delay constraint (10, in this example). It becomes clear that the path along the upper periphery of the network representation would have a delay of 11, which exceeds the delay constraint. The path from the source through nodes 6 and 3, however, has a total path delay of 9. The tuple inside the node circles reflects the sums of link AW and the sums of link delay, respectively.

[0038] From the foregoing, it will be appreciated by those skilled in the art that currently available Dijkstra algorithm technology, including existing program code, can be adapted in accordance with the present inventive teachings to

allow real-time routing at nodes of a data network in response to link data provided in the normal use of PNNI protocols.

[0039] In use of the present inventive methods and systems employing disclosed algorithms and methods, node processors or alternative processor facilities will find one or more feasible paths through a network that satisfy end-to-end delay constraints - if at least one such path exists.

[0040] If the AW shortest-path automatically satisfies the delay constraint, that path will advantageously be chosen using the present inventive teachings. While the above-described methods indicate the use of the Dijkstra algorithm $n + 1$ times for the one-to-all shortest path solution, those skilled in the art will recognize that in appropriate circumstances the computational load can be reduced by exchanging delay-to-destination labels $d_{j,k}$ among nodes. While the present detailed description has proceeded in the context of PNNI protocols, the present invention will prove useful in a variety of network types using a variety of network protocols.

[0041] Other modifications and extensions, all within the spirit and scope of the attached claims, will occur to those skilled in the art in light of the present inventive teachings.

Claims

1. A method for choosing a path in a network from a source to a destination, the network comprising a plurality of nodes, each node connected to a plurality of other nodes by links, each link having a weights reflecting at least first and second link parameters, paths connecting said source and said destination comprising a plurality of links, said paths being subject to a constraint on the cumulative value of said second parameter for links in a path, the method comprising the steps of:

   a. for paths from said destination to said source, performing a minimum-weight path identification based on said second parameter for paths in said network, thereby to generate labels identifying weights for said second parameter for each node *j* to destination *k*,

   b. for paths from said source to said destination performing a minimum-weight path identification from said source using said first parameter as a link metric, and

   c. eliminating any nodes that would cause the constraint on said cumulative value for said second parameter to be violated.

2. The method of claim 1 wherein operation of said network is in accordance with PNNI protocols.

3. The method of claim 1 wherein said first parameter is administrative weight (AW).

4. The method of claim 2 wherein said second parameter is link delay.

5. The method of claim 1 wherein said minimum-weight path identification for paths from said destination is accomplished using a Dijkstra SPF algorithm.

6. The method of claim 5 wherein said minimum-weight path identification for paths from said source is accomplished using a Dijkstra SPF algorithm.

7. The method of claim 6 wherein the method further comprises repeating steps a, b and c for paths from said source node to each possible destination node in said network.

8. The method of claim 7 wherein for destination nodes after a first destination node, path information from said source to one or more other destinations is used in determining acceptable paths to each said nodes after said first destination.

9. The method of claim 1 comprising the further step of eliminating any paths in said minimum-weight path identification for paths from said source for which the cumulative value for said first parameter exceeds a predetermined maximum.

10. The method of claim 1 wherein said step of performing a minimum-weight path identification from said source comprises (i) labeling a node *i* with cumulative AW from the source node, and (ii) tracking cumulative delay ($D_{s,i}$) from the source based on values for said second parameter during said minimum-weight path identification from

said destination.

11. The method of claim 10 wherein said step of eliminating comprises labeling neighboring nodes *j* of a permanently labeled node *i* only if $(D_{s,i} + d_{j,k} + \delta_{i,j})$ is less than the specified end-to-end constraint on said second parameter, where $\delta_{i,j}$ is the value of said second parameter for the link between nodes *i* and *j*, and $d_{j,k}$ is the value for said second parameter for the link between nodes *j* and *k*.

12. The method of claim 10 wherein said second parameter is link delay.

13. The method of claim 11 wherein said second parameter is link delay.

FIG. 1

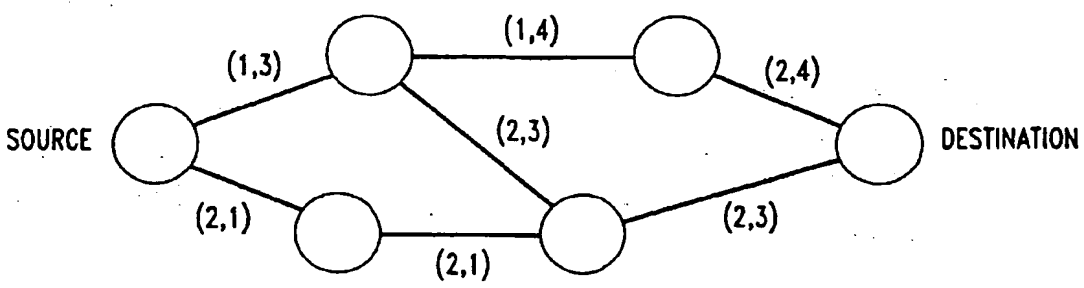


FIG. 2

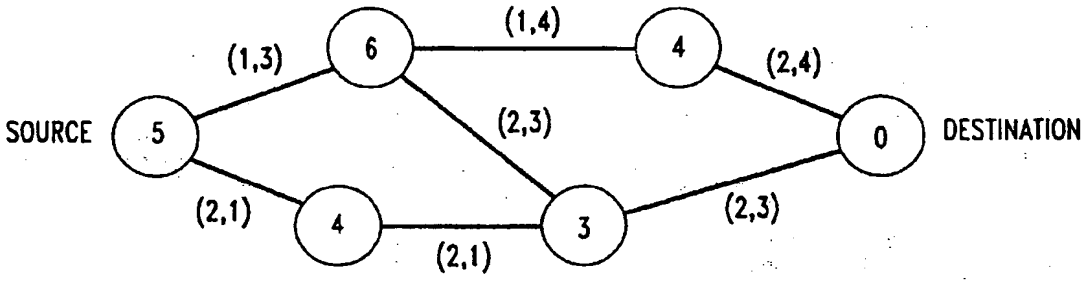
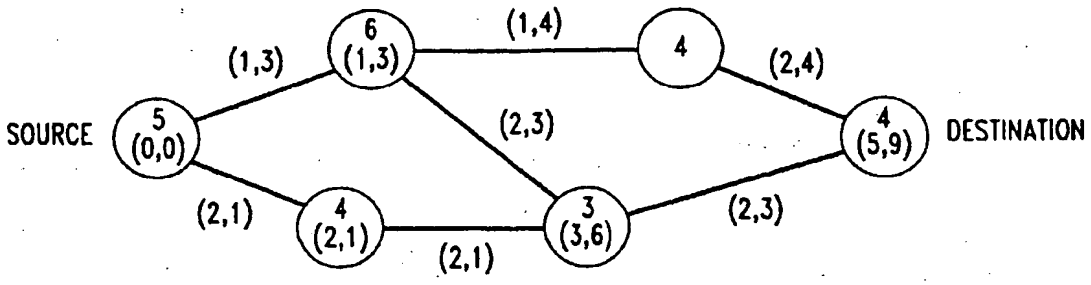


FIG. 3



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 99 30 9837

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	PARSA M ET AL: "AN ITERATIVE ALGORITHM FOR DELAY-CONSTRAINED MINIMUM-COST MULTICASTING" IEEE / ACM TRANSACTIONS ON NETWORKING,US,IEEE INC. NEW YORK, vol. 6, no. 4, 1 August 1998 (1998-08-01), pages 461-474, XP000771977 ISSN: 1063-6692 * page 463, column 1, line 17 - line 26 *	1,5-9	H04L12/56
A	LOGOTHETIS D ET AL: "Delay sensitive routing in PNNI-based ATM networks" IEEE GLOBECOM 1998 (CAT. NO. 98CH36250), IEEE GLOBECOM 1998, SYDNEY,NSW,AUSTRALIA, 8-12 NOV. 1998, pages 604-612 vol.1, XP002133059 1998, Piscataway, NJ, USA, IEEE, USA ISBN: 0-7803-4984-9 * page 604, column 2, line 23 - line 27 * * page 609, column 2, line 27 - line 30 *	1-4	
A	RON WIDYONO: "The design and evaluation of routing algorithms for real-time channels" INTERNET ARTICLE, 'Online! June 1994 (1994-06), pages 1-37, XP002133060 Retrieved from the Internet: <URL:ftp://ftp.icsi.berkeley.edu/pub/techreports/1994/tr-94-024.ps.gz> 'retrieved on 2000-03-14! * page 17 - page 20 *	1	TECHNICAL FIELDS SEARCHED (Int.Cl.7) H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	14 March 2000	Tous Fajardo, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)